The CENTRE for EDUCATION
in MATHEMATICS and COMPUTING

2015 Canadian Computing Competition: Senior Division

Sponsor:

WATERLOO
MATHEMATICS

Canadian Computing Competition
Student Instructions for the Senior Problems

1. You may only compete in one competition. If you wish to write the Junior paper, see the other problem set.

2. Be sure to indicate on your **Student Information Form** that you are competing in the **Senior** competition.

3. You have three (3) hours to complete this competition.

4. You should assume that:

   - if your supervising teacher is grading your solutions, all input is from a file named `sX.in`, where `X` is the problem number ($1 \leq \texttt{X} \leq 5$);
   - if you are using the On-line CCC grader, all input is from standard input;
   - all output is to standard output (i.e., to the screen).

   There is no need for prompting. Be sure your output matches the expected output in terms of order, spacing, etc. IT MUST MATCH EXACTLY!

5. Do your own work. Cheating will be dealt with harshly.

6. Do not use any features that the judge (your teacher or the On-line Grader) will not be able to use while evaluating your programs. In particular, take note of the type and version of the compiler used for your programming language on the On-line Grader if you are using the On-line Grader.

7. Books and written materials are allowed. Any machine-readable materials (like other programs which you have written) are *not* allowed. However, you are allowed to use "standard" libraries for your programming languages; for example, the STL for C++, java.util.*, java.io.*, etc. for Java, and so on.

8. Applications other than editors, compilers, debuggers or other standard programming tools are **not** allowed. Any use of other applications will lead to disqualification.

9. If your teacher is grading, please use file names that are unique to each problem: use `s1.pas` or `s1.c` or `s1.java` (or some other appropriate extension) for Problem S1. If you are using the On-line Grader, follow naming rules described there (and take particular note of file and class names for Java programs).

10. Your program will be run against test cases other than the sample ones. Be sure you test your program on other test cases. Inefficient solutions may lose marks for some problems, especially Problems 4 and 5.

11. The Senior problems shall be given 5 seconds of execution time per test case (if graded in schools), and similarly, 5 seconds of execution time on the on-line grader (if graded using the on-line grader). If this time limit is exceeded, no points will be awarded for that test case.

12. If you finish in the top 20 competitors on this competition, you will be invited to participate in Canadian Computing Olympiad (CCO), held at the University of Waterloo in May 2015. We will select the Canadian International Olympiad in Informatics (IOI) team from among the top contestants at the CCO. You should note that IOI 2015 will be held in Kazakhstan. Note that you will need to know C, C++ or Pascal if you are invited to the CCO. But first, do well on this contest!

13. Check the CCC website at the end of March to see how you did on this contest, and to see who the prize winners are. The CCC website is:

`www.cemc.uwaterloo.ca/contests/computing.html`

Problem S1: Zero That Out

Problem Description
Your boss has asked you to add up a sequence of positive numbers to determine how much money your company made last year.

Unfortunately, your boss reads out numbers incorrectly from time to time.

Fortunately, your boss realizes when an incorrect number is read and says "zero", meaning "ignore the current last number."

Unfortunately, your boss can make repeated mistakes, and says "zero" for each mistake.

For example, your boss may say "One, three, five, four, zero, zero, seven, zero, zero, six", which means the total is 7 as explained in the following chart:

Boss statement(s)	Current numbers	Explanation
"One, three, five, four"	1, 3, 5, 4	Record the first four numbers.
"zero, zero"	1, 3	Ignore the last two numbers.
"seven"	1, 3, 7	Record the number 7 at the end of our list.
"zero, zero"	1	Ignore the last two numbers.
"six"	1, 6	We have read all numbers, and the total is 7.

At any point, your boss will have said at least as many positive numbers as "zero" statements. If all positive numbers have been ignored, the sum is zero.

Write a program that reads the sequence of boss statements and computes the correct sum.

Input Specification
The first line of input contains the integer K ($1 \leq K \leq 100\,000$) which is the number of integers (including "zero") your boss will say. On each of the next K lines, there will either be one integer between 1 and 100 (inclusive), or the integer 0.

Output Specification
The output is one line, containing the integer which is the correct sum of the integers read, taking the "zero" statements into consideration. You can assume that the output will be an integer in the range 0 and $1\,000\,000$ (inclusive).

Sample Input 1
```
4
3
0
4
0
```

Output for Sample Input 1

```
0
```

Sample Input 2

```
10
1
3
5
4
0
0
7
0
0
6
```

Output for Sample Input 2

```
7
```

Problem S2: Jerseys

Problem Description
A school team is trying to assign jerseys numbered $1, 2, 3, \ldots, J$ to student athletes. The size of each jersey is either small (S), medium (M) or large (L).

Each athlete has requested a specific jersey number and a preferred size. The athletes will not be satisfied with a jersey that is the wrong number or that is smaller than their preferred size. They will be satisfied with a jersey that is their preferred size or larger as long as it is the right number. Two students cannot be given the same jersey.

Your task is to determine the maximum number of requests that can be satisfied.

Input Specification
The first line of input is the integer J which is the number of jerseys.
The second line of input is the integer A which is the number of athletes.
The next J lines are each the character S, M or L. Line j gives the size of jersey j $(1 \leq j \leq J)$.
The last A lines are each the character S, M or L followed by a space followed by an integer. Line a $(1 \leq a \leq A)$ gives the requested size and jersey number for athlete a where the athletes are numbered $1, 2, 3, \ldots, A$.

For 50% of the test cases, $1 \leq J \leq 10^3$ and $1 \leq A \leq 10^3$.
For the remaining 50% of the test cases, $1 \leq J \leq 10^6$ and $1 \leq A \leq 10^6$.

Output Specification
The output will consist of a single integer which is the maximum number of requests that can be satisfied.

Sample Input
```
4
3
M
S
S
L
L 3
S 3
L 1
```

Output for Sample Input
```
1
```

Explanation Sample Output
Jersey 1 cannot be assigned because it is medium and athlete 3 requested large. No athlete requested jersey 2 or 4. Jersey 3 (small) can be assigned athlete 2 (small) but not athlete 1 (large).

Problem S3: Gates

Problem Description
For your birthday, you were given an airport.

The airport has G gates, numbered from 1 to G.

P planes arrive at the airport, one after another. You are to assign the ith plane to permanently dock at any gate $1, \ldots, g_i$ ($1 \leq g_i \leq G$), at which no previous plane has docked. As soon as a plane cannot dock at any gate, the airport is shut down and no future planes are allowed to arrive.

In order to keep the person who gave you the airport happy, you would like to maximize the number of planes starting from the beginning that can all dock at different gates.

Input Specification
The first line of input contains G ($1 \leq G \leq 10^5$), the number of gates at the airport.

The second line of input contains P ($1 \leq P \leq 10^5$), the number of planes which will land.

The next P lines contain one integer g_i ($1 \leq g_i \leq G$), such that the ith plane must dock at some gate from 1 to g_i, inclusive.

Note that for at least 40% of the marks for this question, $P \leq 2000$ and $G \leq 2000$.

Output Specification
Output the maximum number of planes that can land starting from the beginning.

Sample Input 1
```
4
3
4
1
1
```

Output for Sample Input 1
```
2
```

Explanation of Output for Sample Input 1
The first plane can go anywhere, but it is best to not put it into Gate 1. Notice that planes 2 and 3 both want to dock into Gate 1, so plane 3 is unable to dock.

Sample Input 2
```
4
6
2
```

```
2
3
3
4
4
```

Output for Sample Input 2

```
3
```

Explanation of Output for Sample Input 2

The first two planes will dock in gates 1 and 2 (in any order). The third plane must dock at Gate 3. Thus, the fourth plane cannot dock anywhere, and the airport is closed, even though plane 5 would have been able to dock.

Problem S4: Convex Hull

Problem Description

You are travelling on a ship in an archipelago. The ship has a convex hull which is K centimetres thick. The archipelago has N islands, numbered from 1 to N. There are M sea routes amongst them, where the ith route runs directly between two different islands a_i and b_i ($1 \leq a_i, b_i \leq N$), takes t_i minutes to travel along in either direction, and has rocks that wear down the ship's hull by h_i centimetres. There may be multiple routes running between a pair of islands.

You would like to travel from island A to a different island B ($1 \leq A, B \leq N$) along a sequence of sea routes, such that your ship's hull remains intact – in other words, such that the sum of the routes' h_i values is strictly less than K.

Additionally, you are in a hurry, so you would like to minimize the amount of time necessary to reach island B from island A. It may not be possible to reach island B from island A, however, either due to insufficient sea routes or the having the ship's hull wear out.

Input Specification

The first line of input contains three integers K, N and M ($1 \leq K \leq 200$, $2 \leq N \leq 2000$, $1 \leq M \leq 10000$), each separated by one space.

The next M lines each contain 4 integers a_i b_i t_i and h_i ($1 \leq a_i, b_i \leq N$, $1 \leq t_i \leq 10^5$, $0 \leq h_i \leq 200$), each separated by one space. The ith line in this set of M lines describes the ith sea route (which runs from island a_i to island b_i, takes t_i minutes and wears down the ship's hull by h_i centimetres). Notice that $a_i \neq b_i$ (that is, the ends of a sea route are distinct islands).

The last line of input contains two integers A and B ($1 \leq A, B \leq N$; $A \neq B$), the islands between which we want to travel.

For 20% of marks for this question, $K = 1$ and $N \leq 200$. For another 20% of the marks for this problem, $K = 1$ and $N \leq 2000$.

Output Specification

Output a single integer: the integer representing the minimal time required to travel from A to B without wearing out the ship's hull, or -1 to indicate that there is no way to travel from A to B without wearing out the ship's hull.

Sample Input 1

```
10 4 7
1 2 4 4
1 3 7 2
3 1 8 1
3 2 2 2
4 2 1 6
3 4 1 1
```

```
1 4 6 12
1 4
```

Output for Sample Input 1

```
7
```

Explanation of Output for Sample Input 1

The path of length 1 from 1 to 4 would wear out the hull of the ship. The three paths of length 2 ($[1, 2, 4]$ and $[1, 3, 4]$ two different ways) take at least 8 minutes. The path $[1, 2, 3, 4]$ takes 7 minutes and only wears down the hull by 7 centimetres, whereas the path $[1, 3, 2, 4]$ takes 13 minutes and wears down the hull by 5 centimetres.

Sample Input 2

```
3 3 3
1 2 5 1
3 2 8 2
1 3 1 3
1 3
```

Output for Sample Input 2

```
-1
```

Explanation of Output for Sample Input 2

The direct path $[1, 3]$ wears down the hull to 0, as does the path $[1, 2, 3]$.

Problem S5: Greedy For Pies

Problem Description
The local pie shop is offering a promotion - all-you-can-eat pies! Obviously, you can't pass up this offer.

The shop lines up N pies from left to right - the ith pie contains A_i grams of sugar. Additionally, another M pies are provided - the ith of these contains B_i grams of sugar.

You are first allowed to insert each of the M pies from the second group anywhere into the first list of N pies, such as at its start or end, or in between any two pies already in the list. The result will be a list of $N + M$ pies with the constraint that the initial N pies are still in their original relative order.

Following this, you are allowed to take one walk along the new line of pies from left to right, to pick up your selection of all-you-can-eat pies! When you arrive at a pie, you may choose to add it to your pile, or skip it. However, because you're required to keep moving, if you pick up a certain pie, you will not be able to also pick up the pie immediately after it (if any). In other words, you cannot eat consecutive pies in this combined list.

Being a pie connoisseur, your goal is to maximize the total amount of sugar in the pies you pick up from the line. How many grams can you get?

Input Specification
The first line of input contains the integer N ($1 \leq N \leq 3000$). The next N lines contain one integer A_i ($1 \leq A_i \leq 10^5$), describing the integer number of grams of sugar in pie i in the group of N pies.

The next line contains M ($0 \leq M \leq 100$), the number of pies in the second list. The next M lines contain one integer B_i ($1 \leq B_i \leq 10^5$), describing the integer number of grams of sugar in pie i in the group of M pies.

For 20% of the marks for this question, $M = 0$. For another 20% of the marks for this question $M = 1$. For another 20% of the marks for this question $M \leq 10$.

Output Specification
Output the maximum number of grams of sugar in all the pies that you are able to pick up.

Sample Input
```
5
10
12
6
14
7
```

```
3
1
8
2
```

Output for Sample Input

```
44
```

Explanation of Output for Sample Input

Place the pies in the order

$$10, 1, 12, 2, 8, 6, 14, 7$$

(that is, insert the pie with 1 gram of sugar between 10 and 12, and insert pies with 2 and 8 grams of sugar, in that order, between pies 12 and 6). Then, we can grab $10 + 12 + 8 + 14 = 44$ grams of sugar, which is maximal.